PANDAS

VON LYNN BÜHL

WAS IST PANDAS?

Pandas ist ein ein schnelles, leistungsstarkes, flexibles und einfach zu bedienendes Open-Source-Tool zur Datenanalyse und -manipulation, das auf der Programmiersprache Python aufbaut.

FUNKTIONEN

- Datenbereinigung
- Datenbefüllung
- Datennormalisierung
- Zusammenführungen und Verknüpfungen
- Datenvisualisierung
- Statistische Analyse
- Dateninspektion
- Daten laden und speichern
- Und vieles mehr

EINLESEN VON DATEIN

```python
import pandas as pd

#Für csv Datein
csv_Datei = pd.read_csv("file_path")
#Für xlsc Datein
xlsx_Datei= pd.read_excel("file_path")

# Zustätliche Tools

# Liest nur die ersten drei Spalten
pd.read_csv("file_path", usecols=[0, 1, 2])

# Überspringt die ersten 3 Zeilen
pd.read_csv("file_path", skiprows=3)

# Verwendet keine Kopfzeile
pd.read_excel("file_path", header=None)

# Legt die Datentypen der Spalten fest
pd.read_excel("file_path", dtype={'column1': str, 'column2': float})
```

Weiter Funktionen: https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

DATAFRAMES IN PANDAS

Was sind DataFrames in Pandas?

- Datenstruktur in Form einer tabellarischen Form mit Zeilen und Spalten
- Indexierung: Jede Zeile in einem DataFrame hat einen eindeutigen Index
- Spaltenname: Jede Spalte hat einen Namen für den Zugriff.
- Datenmanipulation: Bearbeit,Hinzufügen und Löschen von Spalten

DATAFRAMES IN PANDAS

```python
import pandas as pd

#Beispiel Code Tableraische Struktur
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['Berlin', 'Munich', 'Hamburg']}

df = pd.DataFrame(data)
#print(df)

#Beispiel Code Indixierung

# Mit benutzerdefiniertem Index
df = pd.DataFrame(data, index=['ID1', 'ID2', 'ID3'])
#print(df)


#Hinzufügen einer Spalete

# Hinzufügen einer neuen Spalte
df['Telefonnummer'] = ['+49345453', '+453453', '+35753943']
#print(df)
```

Weiter Funktionen: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

SCHREIBEN VON DATEIN

```python
import pandas as pd


path = r"C:\Users\Administrator\Documents\Studium\C++ Kurs\meine_daten.csv"

# Beispiel DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['Berlin', 'Munich', 'Hamburg']}
df = pd.DataFrame(data)

# DataFrame in eine CSV-Datei schreiben
df.to_csv(
    path ,  # Definiert den Dateipfad
    sep=';',            # Legt das Trennzeichen zwischen den Spalten fest (hier: Semikolon)
    header=True,        # Bestimmt, ob Spaltennamen als erste Zeile in die Datei geschrieben werden
    index=False,        # Steuert, ob der Index in die Datei geschrieben werden soll
    mode='w',           # Modus für das Schreiben der Datei ('w' überschreibt eine vorhandene Datei)
    decimal=','         # Definiert das Dezimaltrennzeichen in der CSV-Datei
)
```

Weiter Funktionen: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html

SCHREIBEN VON DATEIEN (NUR IN TO_CSV)

- **mode= ‘w‘ (write)**
- **mode= ‘a‘ (append)**
- **mode= ‘r‘ (read)**
- **mode= ‘r+‘ (read and write)**

Weiter Funktionen: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html

ZUSAMMENFÜGEN VON DATA FRAMES

```
import pandas as pd

# Verkettung von DataFrames entlang der Achse 0 (Zeilen)
concatenated_df = pd.concat([df1, df2, df3], axis=0, ignore_index=True)

# Verkettung von DataFrames entlang der Achse 1 (Spalten)
concatenated_df = pd.concat([df1, df2, df3], axis=1)
```

Weiter Funktionen: https://pandas.pydata.org/pandas-docs/version/0.21/generated/pandas.concat.html

FILTERN VON DATEN

```python
import pandas as pd

# Beispiel DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Alter': [25, 30, 35, 40],
}
df = pd.DataFrame(data)

print(df["Alter"])

# Bedingung: Zeilen filtern, in denen die Spalte 'Wert' größer oder gleich 10 ist
gefiltert_df = df[df['Alter'] <= 30]


print(gefiltert_df)
```

PLOTTEN VON DATAFRAMES

```
import pandas as pd
import matplotlib.pyplot as plt

# Erstelle ein Beispiel DataFrame
data = {'Jahr': [2010, 2011, 2012, 2013, 2014],
        'Umsatz': [500, 600, 750, 900, 1100]}
df = pd.DataFrame(data)

# Erstelle einen Linienplot
df.plot(x='Jahr', y='Umsatz' , kind = "scatter")
plt.title('Umsatz über die Jahre')
plt.xlabel('Jahr')
plt.ylabel('Umsatz')
plt.show()
```

Weiter Funktionen: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html

BEISPIELCODE

```python
import pandas as pd
import matplotlib.pyplot as plt

from faker import Faker
import random

anzahl_zahlen = 100  # Anzahl der zu generierenden Zahlen
zahlen_liste = [random.randint(1, 100) for _ in range(anzahl_zahlen)]
#erstelle Objekt Faker
faker = Faker()


def Generiere():
    namen_liste = []
    adresse_liste = []
    Telefonnummer = []

    for i in range(anzahl_zahlen):
        namen_liste.append(faker.name())
        adresse_liste.append(faker.address())
        Telefonnummer.append(faker.phone_number())

    return namen_liste, adresse_liste, Telefonnummer
```

```python
df1 =pd.DataFrame({"Namen": Generiere()[0] , "Alter": zahlen_liste
                   , "Adresse":  Generiere()[1]})

df1["Telefonnummer"] =  Generiere()[2]

#print(df1)

df2 =pd.DataFrame({"Namen": Generiere()[0] , "Alter": zahlen_liste ,
                   "Adresse":  Generiere()[1], "Telefonnummer" : Generiere()[2]})
#print(df2)

concatenated_df = pd.concat([df1, df2], axis=0)

#print(concatenated_df)

concatenated_df = concatenated_df[(concatenated_df["Alter"] > 25)
                                  & (concatenated_df["Alter"] <= 30)]
#print(concatenated_df)

path = r"C:\Users\Administrator\Documents\Studium\C++ Kurs\Namen.xlsx"
concatenated_df.to_excel(path)

concatenated_df['Alter'].plot(kind='hist')
plt.xlabel('Alter')
plt.ylabel('Anzahl der Personen')
plt.show()
```

QUELLEN

- https://pandas.pydata.org/
- https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/
- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html